5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/728,441	12/01/2000	Wen-mei W. Hwu	042302 0269900	3226

27498 7590 06/29/2007

PILLSBURY WINTHROP SHAW PITTMAN LLP
P.O. BOX 10500
MCLEAN, VA 22102

EXAMINER	
LI, AIMEE J	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
06/29/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/728,441	HWU ET AL.	
	Examiner	Art Unit	
	Aimee J. Li	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 April 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-50,53 and 54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *1-25,32-34 and 54* is/are allowed.

6)☒ Claim(s) *26-31,35-50 and 53* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-50, 53, and 54 have been considered. Claims 1, 32, 35, and 43 have been amended as per Applicant's request.

Allowable Subject Matter

2. Claims 1-25, 32-34, and 54 are allowed.

3. The reasons for allowance for claims 32-34 were stated in the previous Office Action.

4. The following is an examiner's statement of reasons for allowance for claims 1-25 and 54: Independent claim 1 recites the limitations for storing scheduling information which defines for each cycle whether and which stored instructions are executed by the functional unit in that cycle in a buffer in the dispatch stage that also stores the instructions. This has not been taught in the prior art search and found nor has the prior art that teaches individual limitations provided adequate reasons to combine. The reasons to combine would also not be apparent to one of ordinary skill in the art due to the complexities of implementing the combinations.

5. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 26-31, 35-41, 43-49, and 53 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah, U.S. Patent Number 5,989,865 (herein referred to as Mahalingaiah) in view of George, U.S. Patent Number 4,626,988 (herein referred to as George) and in further view of Subramanian et al., U.S. Patent Number 5,867,711 (herein referred to as Subramanian).

8. Referring to claim 26, Mahalingaiah has taught a buffer in the dispatch stage of a processor, the buffer being associated with a functional unit that executes instructions issued to it from the dispatch stage (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; Figure 1; column 17, line 66 to column 18, line 16; and Figure 4). Mahalingaiah has not taught a first portion for receiving from a fetch stage and storing a kernel set of loop instructions. George has taught a first portion for receiving from a fetch stage and storing a kernel set of loop instructions (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68). A person of ordinary skill in the art at the time the invention was made, and as taught by George, would have recognized that the loop buffer eliminates redundancy, reduces storage contention, and providing efficient management for loop mode operations (George column 1, lines 26-47 and column 2, lines 57-68). Therefore, it would have been obvious to a person of ordinary skill in the at the time the invention was made to incorporate the loop buffer of George in the device of Mahalingaiah to eliminate redundancy, reduce storage contention, and provide efficient management for loop mode operations. In addition, Mahalingaiah has not taught a plurality of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, wherein the instructions are issued to the functional unit from the buffer in accordance with the stored modulo schedule stage identifiers so as to cause the functional unit to execute a number of

iterations of a loop. Subramanian has taught a plurality of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, wherein the instructions are issued to the functional unit from the buffer in accordance with the stored modulo schedule stage identifiers so as to cause the functional unit to execute a number of iterations of a loop (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5). In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

9. Referring to claims 27-31, Mahalingaiah has taught

   a. Wherein the processor further includes control logic, the buffer being coupled to the control logic and the functional unit for causing the kernel set of loop instructions to be issued to the functional unit (Applicant's claim 27) (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4);

   b. Wherein the loop instructions comprise undecoded instructions, and wherein the processor further includes a decode stage interposed between the functional unit

and the buffer for decoding the instructions (Applicant's claim 30) (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; Figure 1; column 18, lines 50-57; and Figure 4); and

c. Wherein the loop instructions comprise decoded instructions in the form of functional unit control signals (Applicant's claim 31) (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; and Figure 1). In regards to Mahalingaiah, instructions are functional unit control signals since they control how the functional unit operates.

10. Mahalingaiah has not taught

a. The kernel set of loop instructions and modulo schedule stage identifiers (Applicant's claim 27);

b. Wherein the modulo schedule stage identifiers comprise bit fields (Applicant's claim 28);

c. The control logic determining whether to issue a certain one of the loop instructions to the functional unit in accordance with a bit position of a set bit in the bit field corresponding to the certain loop instruction (Applicant's claim 28); and

d. Wherein the control logic is operative to cause a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions in accordance with the stored modulo schedule stage identifiers (Applicant's claim 29).

11. Subramanian has taught:

a. The kernel set of loop instructions and modulo schedule stage identifiers (Applicant's claim 27) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5);

b. Wherein the modulo schedule stage identifiers comprise bit fields (Applicant's claim 28) (Subramanian column 3, lines 36-44);

c. The control logic determining whether to issue a certain one of the loop instructions to the functional unit in accordance with a bit position of a set bit in the bit field corresponding to the certain loop instruction (Applicant's claim 28) (Subramanian column 2, lines 10-16; column 4, lines 16-26; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5); and

d. Wherein the control logic is operative to cause a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions in accordance with the stored modulo schedule stage identifiers (Applicant's claim 29) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 6, lines 4-19; column 10, lines 5-9; Figure 4; and Figure 5).

12. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped

(Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

13. Referring to claims 35 and 43, Mahalingaiah has taught a method for executing a number of iterations of a loop in a processor, the method comprising:

a. Storing the kernel set of loop instructions at a dispatch stage of the processor (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4); and

b. Storing loop parameters in control logic associated with the stored loop instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

14. Mahalingaiah has not taught receiving the kernel set of loop instructions and loop parameters from an instruction stream fetched by the processor. George has taught receiving the kernel set of loop instructions and loop parameters from an instruction stream fetched by the processor (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68). A person of ordinary skill in the art at the time the invention was made, and as taught by George, would have recognized that the loop buffer eliminates redundancy, reduces storage contention, and providing efficient management for loop mode operations (George column 1, lines 26-47 and column 2, lines 57-68). Therefore, it would have been obvious to a person of ordinary skill in the at the time the invention was made to incorporate the loop buffer of George in the device

of Mahalingaiah to eliminate redundancy, reduce storage contention, and provide efficient management for loop mode operations.

15. In addition, Mahalingaiah has not taught:

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions; and

b. Causing the stored kernel set of instructions to be selectively issued to functional units of the processor in accordance with the stored loop parameters so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions.

16. Subramanian has taught:

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions (Subramanian column 6, lines 4-19 and Figure 5); and

b. Causing the stored kernel set of instructions to be selectively issued to functional units of the processor in accordance with the stored loop parameters so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

17. In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the

invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

18. Referring to claims 36-40 and 44-48, Mahalingaiah has taught

    a. A loop iteration register for storing a loop iteration parameter (Applicant's claims 36 and 44) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7), and

    b. A loop cycles register for storing a loop cycles parameter (Applicant's claims 36 and 44) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7);

19. Mahalingaiah has not taught:

    a. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 36 and 44);

    b. Adjusting the value in the loop cycles register from the stored loop cycles parameter in accordance with a processor cycle (Applicant's claims 37 and 45);

    c. Resetting the value in the loop cycles register in accordance with the adjusted value and the stored iteration initiation parameter (Applicant's claims 37 and 45);

d. Adjusting the value in the loop iteration register in accordance with the resetting step (Applicant's claims 38 and 46);

e. Completing the issue of loop instructions in accordance with the adjusted value in the loop iteration register (Applicant's claims 38 and 46);

f. Storing a set of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, the step of causing the stored kernel set of loop instructions to be selectively issued including the step of comparing the stored loop parameters with the modulo schedule stage identifiers (Applicant's claims 39 and 47); and

g. Wherein the step of comparing the stored loop parameters with the modulo schedule stage identifiers includes the step of indexing to a certain one of the modulo schedule stage identifiers in accordance with a current cycle in the modulo schedule stage indicated by the stored loop parameters, the step of causing the stored kernel set of loop instructions to be selectively issued further including issuing the associated loop instruction to the functional unit if the certain modulo schedule stage identifier indicates that the functional unit is active (Applicant's claims 40 and 48).

20. Subramanian has taught:

a. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 36 and 44) (Subramanian column 5, lines 49-56 and Figure 5);

b. Adjusting the value in the loop cycles register from the stored loop cycles parameter in accordance with a processor cycle (Applicant's claims 37 and 45) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

c. Resetting the value in the loop cycles register in accordance with the adjusted value and the stored iteration initiation parameter (Applicant's claims 37 and 45) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

d. Adjusting the value in the loop iteration register in accordance with the resetting step (Applicant's claims 38 and 46) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

e. Completing the issue of loop instructions in accordance with the adjusted value in the loop iteration register (Applicant's claims 38 and 46) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

f. Storing a set of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, the step of causing the stored kernel set of loop instructions to be selectively issued including the step of comparing the stored loop parameters with the modulo schedule stage identifiers (Applicant's claims 39 and 47) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5); and

g. Wherein the step of comparing the stored loop parameters with the modulo schedule stage identifiers includes the step of indexing to a certain one of the modulo schedule stage identifiers in accordance with a current cycle in the modulo schedule stage indicated by the stored loop parameters, the step of

causing the stored kernel set of loop instructions to be selectively issued further including issuing the associated loop instruction to the functional unit if the certain modulo schedule stage identifier indicates that the functional unit is active (Applicant's claims 40 and 48) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5).

21. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

22. Referring to claims 41 and 49, Mahalingaiah has taught shutting down the fetch unit during execution of the number of iterations of the loop (Mahalingaiah column 18, lines 58-60).

23. Referring to claim 53, Mahalingaiah has taught a processor comprising:

a. A functional unit (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; and Figure 1);

b. A buffer that is coupled to provide instructions for execution by the functional unit (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4);

c. Control logic coupled to the buffer (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4).

24. Mahalingaiah has not taught

a. Receive loop parameter associated with loop instructions issued to the functional unit from a fetch stage;

b. Cause a kernel set of the loop instructions issued to the functional unit to be stored in the buffer in accordance with the received loop parameters.

25. George has taught receiving

a. Receive loop parameter associated with loop instructions issued to the functional unit from a fetch stage (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68);

b. Cause a kernel set of the loop instructions issued to the functional unit to be stored in the buffer in accordance with the received loop parameters (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68).

26. A person of ordinary skill in the art at the time the invention was made, and as taught by George, would have recognized that the loop buffer eliminates redundancy, reduces storage contention, and providing efficient management for loop mode operations (George column 1, lines 26-47 and column 2, lines 57-68). Therefore, it would have been obvious to a person of ordinary skill in the at the time the invention was made to incorporate the loop buffer of George in the device of Mahalingaiah to eliminate redundancy, reduce storage contention, and provide efficient management for loop mode operations.

27. In addition, Mahalingaiah has not taught

a. Cause the functional unit to execute a number of iterations of the kernel set of the loop instructions based on the received loop parameter;

b. Cause the functional unit to further execute a prologue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the receive loop parameters; and

c. Cause the functional unit to further execute an epilogue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the received loop parameters.

28. Subramanian has taught

a. Cause the functional unit to execute a number of iterations of the kernel set of the loop instructions based on the received loop parameter (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5);

b. Cause the functional unit to further execute a prologue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the receive loop parameters (Subramanian column 6, lines 4-19 and Figure 5); and

c. Cause the functional unit to further execute an epilogue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the received loop parameters (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

29. In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

30. Claims 42 and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah in view of Subramanian as applied to claims 35 and 43 above, and further in view of Mason et al., U.S. Patent Number 6,418,489 (herein referred to as Mason). Mahalingaiah has not taught wherein the control logic is further operative to allow interrupts to be handled at the end of a current loop iteration. Mason has taught wherein the control logic is further operative to allow interrupts to be handled at the end of a current loop iteration and to complete the number of loop iterations after the interrupt is handled (Mason column 6, lines 32-33). In regards to Mason, returning to complete the loop iterations is inherent to interrupts, since they are only temporary. Please see InstantWeb's Online Computing Dictionary. A person of ordinary skill in the art at the time the invention was made would have recognized that waiting until the end of a loop iteration to allow interrupts would ensure data is not lost and/or corrupted and continuing afterwards ensures the process completes and normal process has resumed. Therefore, it would

have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the interrupts of Mason in Mahalingaiah.

Response to Arguments

31. Applicant's arguments filed 06 April 2007 have been fully considered but they are not persuasive.

32. Applicant argues in essence on pages 21-22

> ...It would not suggest hardware that stores modulo schedule stage identifiers associated with instructions that are used to cause the instructions to be selectively issued to a functional unit...

33. This has not been found persuasive. It seems that Applicants are arguing that, since Subramanian teaches his invention in compiler, i.e. software, and ignoring that the functionality is similar, the claim limitations are not met. However, whether the functionality is implemented in hardware or software, it does not matter. As Tanenbaum's Structured Computer Organization ©1984 has taught, hardware and software are logically equivalent and it is a design decision. Therefore, it is not a patentable function and, whether a patent teaches the functionality in hardware or software, it does not matter.

34. Applicant argues on page 22

> ...Rather, Subramanian requires explicitly defining the prologue and epilogue sets of instructions separately from the kernel set of instructions...

35. This has not been found persuasive. It is unclear how this argument pertains to the cited portion of the claim. In fact, since Subramanian identifies the prologue and epilogues sets of instructions, this means that the three separate sets are identified and stored. Then, as Mahalingaiah teaches, the instructions are executed. This means that the claim limitations are met.

36. Applicant argues on page 22

> ...Nowhere does George suggest receiving loop parameters as explicitly required by the claims.

37. This has not found persuasive. The loop size calculated in George is a loop parameter. There is nothing in the claim language to suggest that the loop size is not a loop parameter. In fact, there is nothing in the claim language to define "loop parameter". It would be improper to import the limitations in the specification of "loop parameter". In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., loop parameters definitions) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

38. Applicant argues on pages 24-25

> ...the cited prior art does not suggest the explicit limitations in claim 53...

39. This has not been found persuasive. It is unclear what the arguments towards this end are. All the arguments have is a mere assertion that the limitations are not taught without evidence or explanation towards how and why they are not taught. As such, all the Examiner can respond with is re-asserting the rejection above. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Conclusion

40. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

41. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

42. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J. Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:00am-4:30pm.

43. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

44. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would

like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Aimee J Li
Examiner
Art Unit 2183

24 June 2007

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100